Program No: 01:

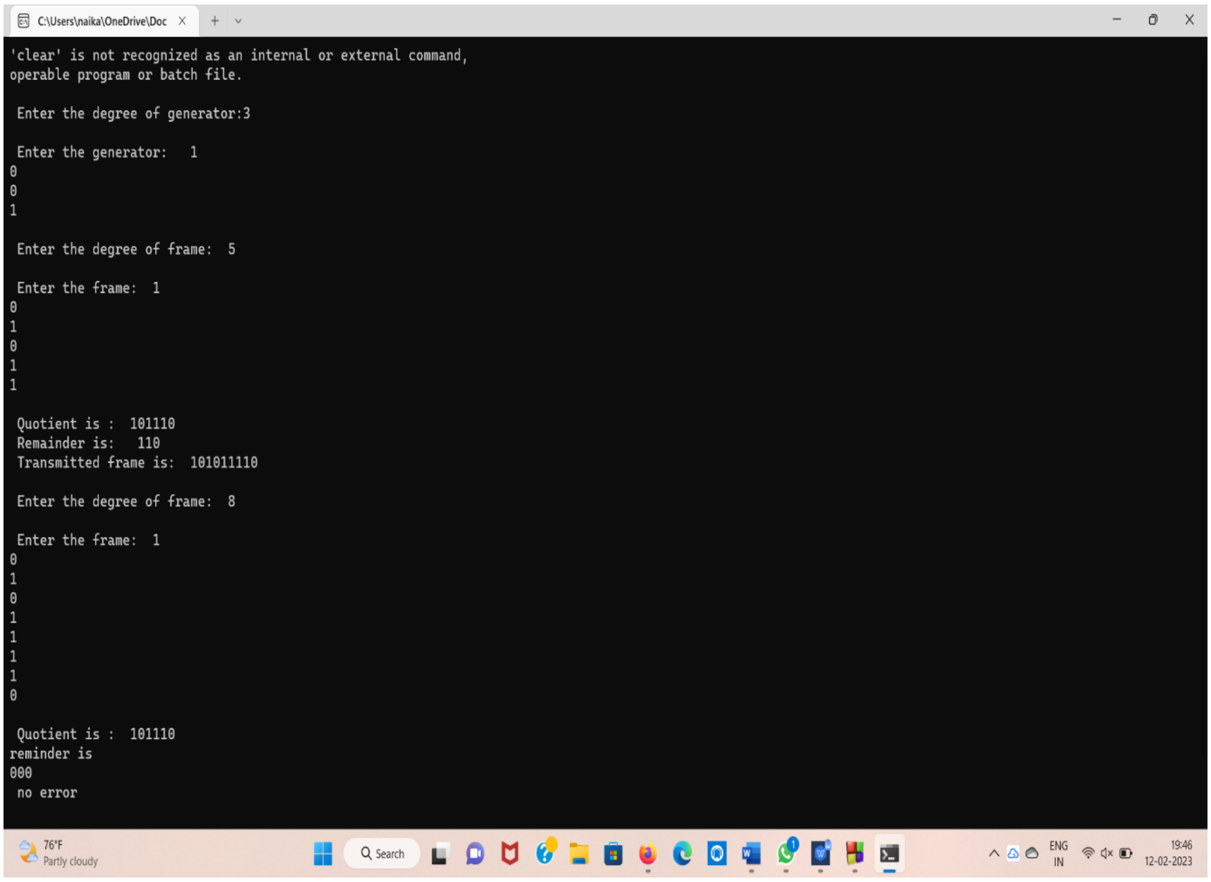
Write a program for error detecting code using CRC-CCITT (16-bits).

```
#include<stdio.h>

int main()

{

	int i,j,k=0;

	int flag=1,a[16],g[16],r[20],div[16],n,m;

	system("clear");

	printf("\n Enter the degree of generator");

	scanf("%d",&n);

	printf("\n Enter the generator:   ");

	for(i=0;i<=n;i++)

	scanf("%d",&g[i]);

	printf("\n Enter the degree of frame:  ");

	scanf("%d",&m);

	printf("\n Enter the frame:  ");

	for(i=0;i<=m;i++)

	scanf("%d",&a[i]);

	if(m<n || (g[0]&&g[n])==0)

	{
        printf("not a proper generator\n");

        exit(0);

}

for(i=m+1;i<=m+n;i++)

a[i]=0;

for(j=0;j<=n;j++)

r[j]=a[j];

for(i=n;i<=m+n;i++)

{

        if(i>n)

        {

                for(j=0;j<n;j++)

                r[j]=r[j+1];

                r[j]=a[i];

        }

        if(r[0])

                div[k++]=1;

        else

        {

                div[k++]=0;
```

```
                    continue;

          }

          for(j=0;j<=n;j++)

                    r[j]=r[j]^g[j];

}



printf("\n Quotient is :  ");

for(j=0;j<k;j++)

printf("%d",div[j]);

printf("\n Remainder is:   ");

for(i=1;i<=n;i++)

printf("%d",r[i]);

printf("\n Transmitted frame is:  ");

for(i=m+1,j=1;i<=m+n;i++,j++)

a[i]=r[j];

for(i=0;i<=m+n;i++)

printf("%d",a[i]);

printf("\n");

printf("\n Enter the degree of frame:  ");

scanf("%d",&m);
```

```
printf("\n Enter the frame:  ");

for(i=0;i<=m;i++)

scanf("%d",&a[i]);

for(j=0;j<=n;j++)

r[j]=a[j];

k=0;

for(i=n;i<=m;i++)

{

        if(i>n)

        {

                for(j=0;j<n;j++)

                        r[j]=r[j+1];

                r[j]=a[i];

        }

        if(r[0])

                div[k++]=1;

        else

        {

                div[k++]=0;

                continue;
                }
                for(j=0;j<=n;j++)
                        r[j]=r[j]^g[j];
        }
                printf("\n Quotient is :  ");
        for(j=0;j<k;j++)
        printf("%d",div[j]);
        printf("\nreminder is\n");
        for(i=1;i<=n;i++)
        printf("%d",r[i]);
        for(i=1;i<=n;i++)
        {
                if(r[i])
                flag=0;
        }
        if(flag)
                printf("\n no error\n");
        else
                printf("\n error\n");
}
```

OUTPUT :



```
'clear' is not recognized as an internal or external command,
operable program or batch file.

 Enter the degree of generator:3

 Enter the generator:   1
0
0
1

 Enter the degree of frame:  5

 Enter the frame:  1
0
1
0
1
1

 Quotient is :  101110
 Remainder is:   110
 Transmitted frame is:  101011110

 Enter the degree of frame:  8

 Enter the frame:  1
0
1
0
1
1
1
1
0

 Quotient is :  101110
reminder is
000
 no error
```

Program No: 02:

Write a program for frame sorting technique used in buffers.

```
#include<stdio.h>

#include<string.h>

struct frame

{

	int seq;

	int len;

	int flag;

	char data[10];

} n[20],m[20],temp;

char str[100];

int count=0;

void frames()

{

	int i,j,s,size,total=0;

	s=strlen(str);

	while(total<s)

	{

		size=rand()%10+1;
```

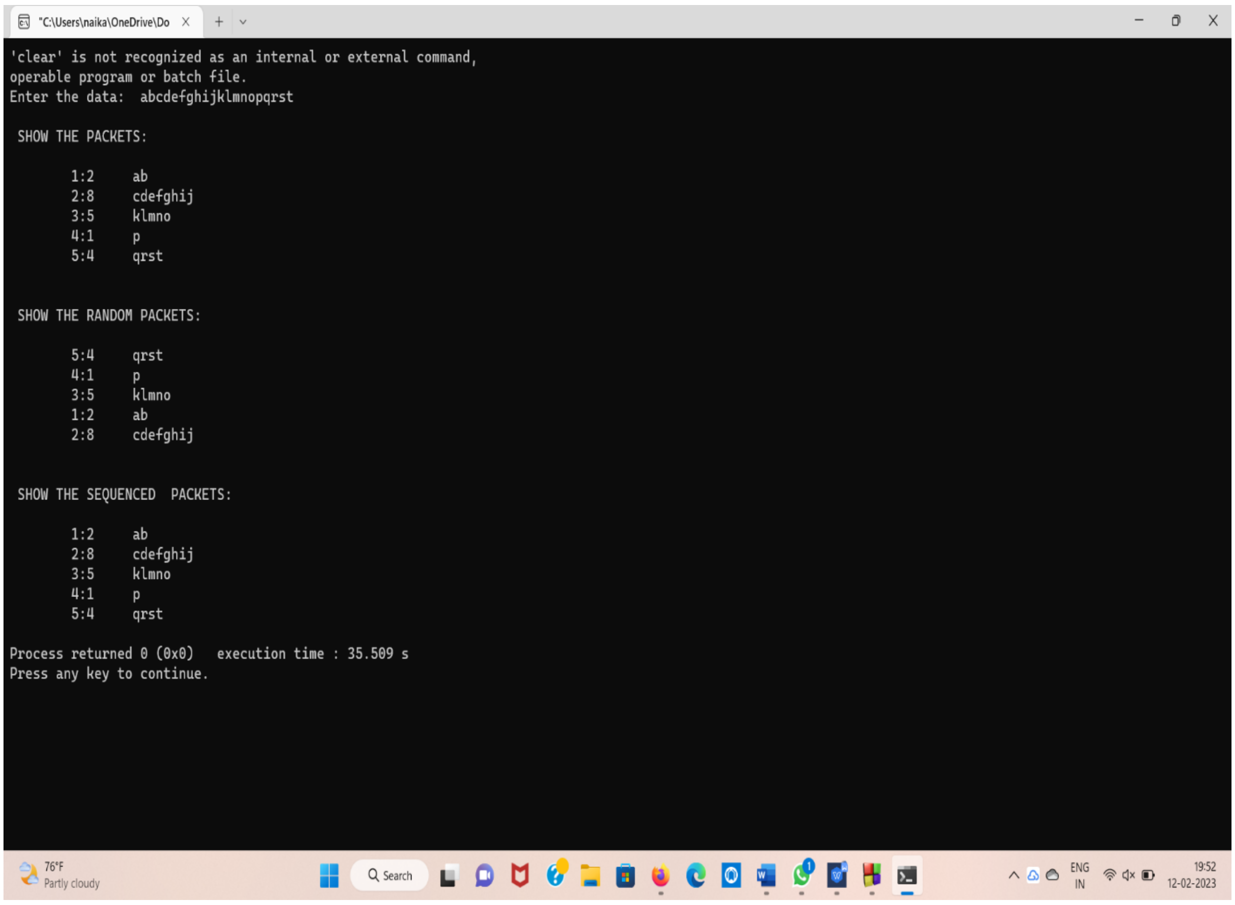
```
        n[count].seq=count+1;

        n[count].len=size;

        n[count].flag=0;



        if((total+size) < s)

        {

                for(i=total,j=0;j<size;i++,j++)

                        n[count].data[j]=str[i];

                total+=size;

        }

        else

        {

                n[count].len=s-total;

                for(j=0;j<n[count].len;j++)

                        n[count].data[j]=str[total++];

        }

        count+=1;

}

printf("\n SHOW THE PACKETS:\n\n");

for(i=0;i<count;i++)
```

```
}
```

```
	{
		printf("\t%d:%d\t",n[i].seq,n[i].len);
		for(j=0;j<n[i].len;j++)
			printf("%c",n[i].data[j]);
		printf("\n");
	}
}
void trans()
{
	int i,j;
	int c=0;
	while(c<count)
	{
		i=rand()%count;
		if(n[i].flag==0)
		{
			m[c++]=n[i];
			n[i].flag=1;
		}
	}
	printf("\n\n SHOW THE RANDOM PACKETS:\n\n");

	for(i=0;i<count;i++)

	{

		printf("\t%d:%d\t", m[i].seq, m[i].len);

		for(j=0;j<m[i].len;j++)

			printf("%c", m[i].data[j]);

		printf("\n");

	}

}


void sort()

{

	int i,j;

	for(i=0;i<count;i++)

		for(j=i+1;j<count;j++)

			if(m[i].seq>m[j].seq)

			{

				temp=m[i];

				m[i]=m[j];

				m[j]=temp;
```

```
                        }

        printf("\n\n SHOW THE SEQUENCED  PACKETS:\n\n");

        for(i=0;i<count;i++)

        {

                printf("\t%d:%d\t",m[i].seq,m[i].len);

                for(j=0;j<m[i].len;j++)

                        printf("%c",m[i].data[j]);

                printf("\n");

        }

}



main()

{       system("clear");

        printf("Enter the data:  ");

        scanf("%s",(str));

        frames();

        trans();

        sort();

}
```

OUTPUT :



```
'clear' is not recognized as an internal or external command,
operable program or batch file.
Enter the data:  abcdefghijklmnopqrst

 SHOW THE PACKETS:

        1:2     ab
        2:8     cdefghij
        3:5     klmno
        4:1     p
        5:4     qrst


 SHOW THE RANDOM PACKETS:

        5:4     qrst
        4:1     p
        3:5     klmno
        1:2     ab
        2:8     cdefghij


 SHOW THE SEQUENCED  PACKETS:

        1:2     ab
        2:8     cdefghij
        3:5     klmno
        4:1     p
        5:4     qrst

Process returned 0 (0x0)   execution time : 35.509 s
Press any key to continue.
```

Program No: 03:

Write a program for bellman-ford algorithm to find shortest path between vertices.

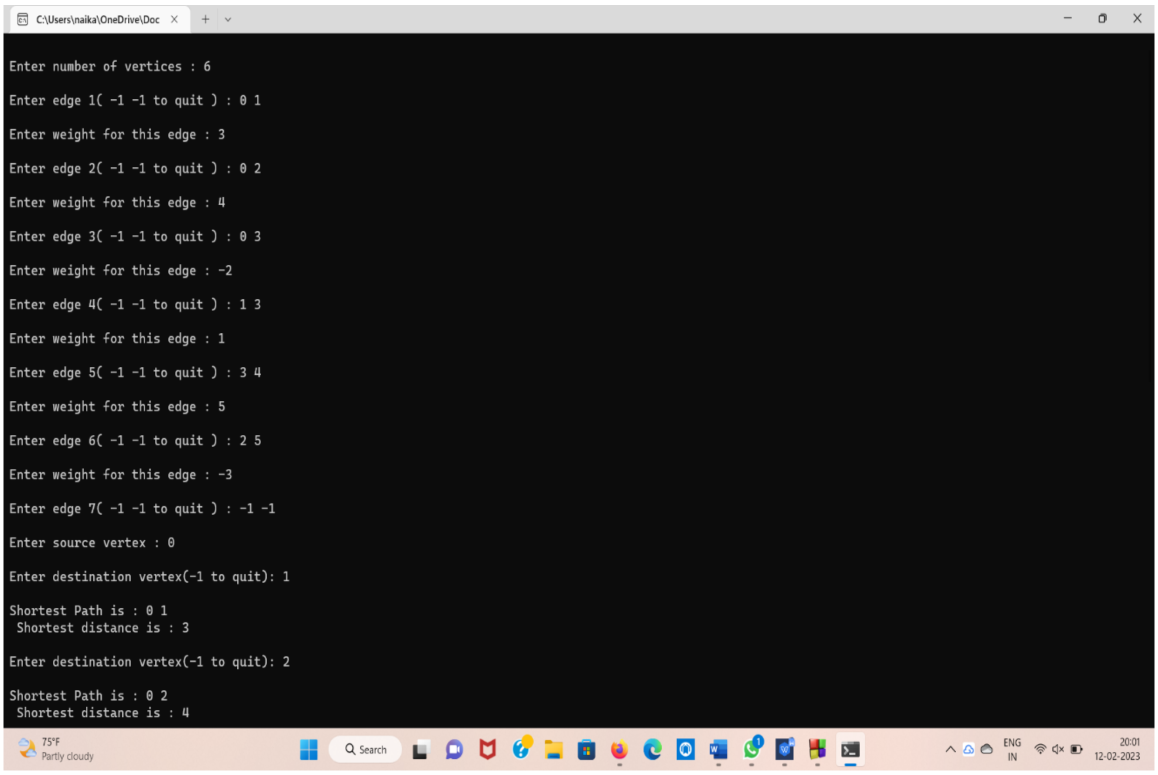
```
#include<stdio.h>
#include<stdlib.h>

#define MAX 100
#define infinity 9999
#define NIL -1
#define TRUE 1
#define FALSE 0

int n; /*Number of vertices in the graph*/
int adj[MAX][MAX]; /*Adjacency Matrix*/

int predecessor[MAX];
int pathLength[MAX];
int isPresent_in_queue[MAX];

int front,rear;
int queue[MAX];
void initialize_queue( );
void insert_queue(int u);
int delete_queue();
int isEmpty_queue();
void create_graph( );
void findPath(int s, int v);
int BellmanFord(int s);

int main()
{
    int flag,s,v;
```

```
        create_graph();

        printf("\nEnter source vertex : ");
        scanf("%d",&s);

        flag = BellmanFord(s);

        if(flag == -1)
        {
                printf("\nError : negative cycle in Graph\n");
                exit(1);
        }

        while(1)
        {
                printf("\nEnter destination vertex(-1 to quit): ");
                scanf("%d",&v);
                if(v == -1)
                        break;
                if(v < 0 || v >= n )
                        printf("\nThis vertex does not exist\n");
                else if(v == s)
                        printf("\nSource and destination vertices are same\n");
                else if( pathLength[v] == infinity )
            printf("\nThere is no path from source to destination vertex\n");
                else
                        findPath(s,v);
        }
        return 0;
}/*End of main()*/
```

```
void findPath(int s, int v )
{
        int i,u;
        int path[MAX]; /*stores the shortest path*/
        int shortdist = 0; /*length of shortest path*/
        int count = 0; /*number of vertices in the shortest path*/

        /*Store the full path in the array path*/
        while( v != s )
        {
                count++;
                path[count] = v;
                u = predecessor[v];
                shortdist += adj[u][v];
                v = u;
        }
        count++;
        path[count]=s;

        printf("\nShortest Path is : ");
        for(i=count; i>=1; i--)
                printf("%d ",path[i]);
        printf("\n Shortest distance is : %d\n", shortdist);
}/*End of findPath()*/

int BellmanFord(int s)
{
        int k = 0,i,current;

        for(i=0;i<n;i++)
        {
                predecessor[i] = NIL;
                pathLength[i] = infinity;
                isPresent_in_queue[i] = FALSE;
        }

        initialize_queue( );
        pathLength[s] = 0; /*Make pathLength of source vertex 0*/
        insert_queue(s); /*Insert the source vertex in the queue*/
      isPresent_in_queue[s] = TRUE;
        while( !isEmpty_queue( ) )
        {
                current = delete_queue( );
                isPresent_in_queue[current] = FALSE;
                if(s == current)
                        k++;
                if(k > n )
                        return -1;/*Negative cycle reachable from source vertex*/
                for(i=0;i<n;i++)
                {
                        if ( adj[current][i] != 0 )
                                if( pathLength[i] > pathLength[current] + adj[current][i] )
                                {
                                        pathLength[i] = pathLength[current] + adj[current][i];
                                        predecessor[i] = current;
                                        if( !isPresent_in_queue[i] )
                                        {
                                                insert_queue(i);
                                                isPresent_in_queue[i]=TRUE;
                                        }
                                }
                }
        }
        return 1;
}/*End of BellmanFord()*/

void initialize_queue( )
{
        int i;
        for(i=0;i<MAX;i++)
                queue[i] = 0;
        rear = -1;front = -1;
}/*End of initailize_queue()*/

int isEmpty_queue()
{
        if(front == -1 || front>rear )
                return 1;
        else
                return 0;
 }/*End of isEmpty_queue()*/

void insert_queue(int added_item)
{
        if (rear == MAX-1)
        {
                printf("\nQueue Overflow\n");
                exit(1);
        }
        else
        {
                if (front == -1) /*If queue is initially empty */
                        front = 0;
                rear = rear+1;
                queue[rear] = added_item ;
        }
}/*End of insert_queue()*/

int delete_queue()
{
        int d;
        if (front == -1 || front > rear)
	{
		printf("\nQueue Underflow\n");
		exit(1);
	}
	else
	{
		d = queue[front];
		front=front+1;
	}
	return d;
}/*End of delete_queue() */



void create_graph()
{
	int i,max_edges,origin,destin, wt;

	printf("\nEnter number of vertices : ");
	scanf("%d",&n);
	max_edges=n*(n-1);

	for(i=1;i<=max_edges;i++)
	{
		printf("\nEnter edge %d( -1 -1 to quit ) : ",i);
		scanf("%d %d",&origin,&destin);

		if( (origin == -1) && (destin == -1) )
			break;

		printf("\nEnter weight for this edge : ");
		scanf("%d",&wt);

		if( origin >= n || destin >= n || origin<0 || destin<0)
```

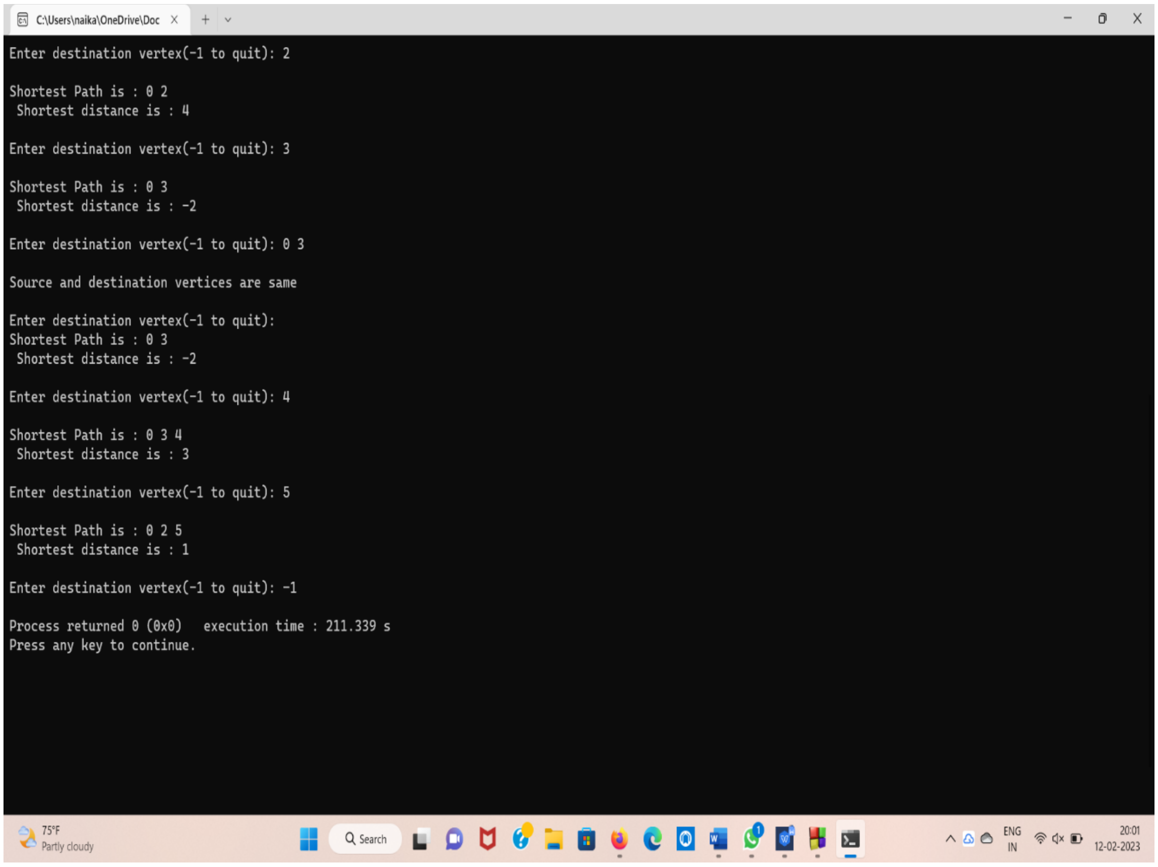
```
        {
                printf("\nInvalid edge!\n");
                i--;
        }
        else
                adj[origin][destin] = wt;
}
```

OUTPUT :



C:\Users\naika\OneDrive\Doc

```
Enter number of vertices : 6

Enter edge 1( -1 -1 to quit ) : 0 1

Enter weight for this edge : 3

Enter edge 2( -1 -1 to quit ) : 0 2

Enter weight for this edge : 4

Enter edge 3( -1 -1 to quit ) : 0 3

Enter weight for this edge : -2

Enter edge 4( -1 -1 to quit ) : 1 3

Enter weight for this edge : 1

Enter edge 5( -1 -1 to quit ) : 3 4

Enter weight for this edge : 5

Enter edge 6( -1 -1 to quit ) : 2 5

Enter weight for this edge : -3

Enter edge 7( -1 -1 to quit ) : -1 -1

Enter source vertex : 0

Enter destination vertex(-1 to quit): 1

Shortest Path is : 0 1
 Shortest distance is : 3

Enter destination vertex(-1 to quit): 2

Shortest Path is : 0 2
 Shortest distance is : 4
```



C:\Users\naika\OneDrive\Doc

```
Enter destination vertex(-1 to quit): 2

Shortest Path is : 0 2
 Shortest distance is : 4

Enter destination vertex(-1 to quit): 3

Shortest Path is : 0 3
 Shortest distance is : -2

Enter destination vertex(-1 to quit): 0 3

Source and destination vertices are same

Enter destination vertex(-1 to quit):
Shortest Path is : 0 3
 Shortest distance is : -2

Enter destination vertex(-1 to quit): 4

Shortest Path is : 0 3 4
 Shortest distance is : 3

Enter destination vertex(-1 to quit): 5

Shortest Path is : 0 2 5
 Shortest distance is : 1

Enter destination vertex(-1 to quit): -1

Process returned 0 (0x0)   execution time : 211.339 s
Press any key to continue.
```

}

<u>Program No: 04:</u>

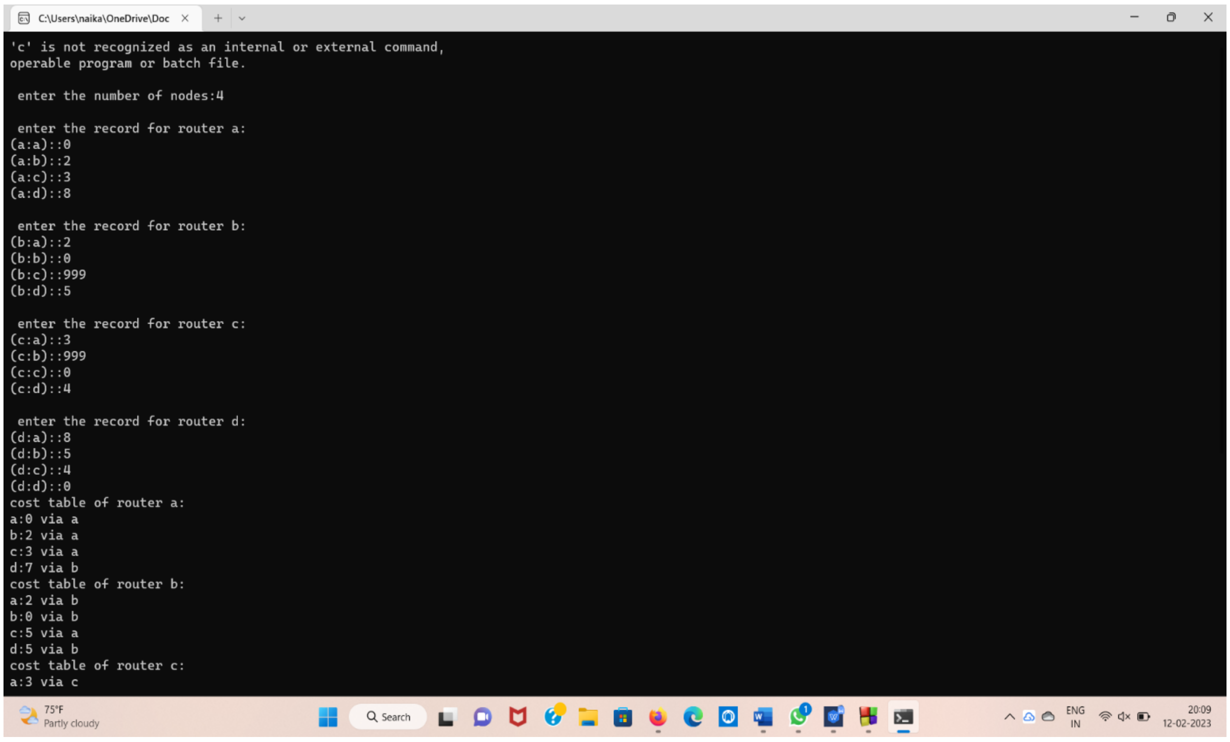
Write a program for distance vector algorithm to find suitable path for transmission

```
#include<stdio.h>

#include<stdlib.h>

int d[10][10],via[10][10];

int main()

{

	int i,c,j,k,n,m,cost,g[10][10],temp[10][10],ch,count;

	system("c");

	printf("\n enter the number of nodes:");

	scanf("%d",&n);

	for(i=0;i<n;i++)

	{

		printf("\n enter the record for router %c:\n",i+97);

		for(j=0;j<n;j++)

		{

			printf("(%c:%c)::",i+97,j+97);

			scanf("%d",&g[i][j]);

			if(g[i][j]!=999)

				d[i][j]=1;

		}

	}

	for(i=0;i<n;i++)
```

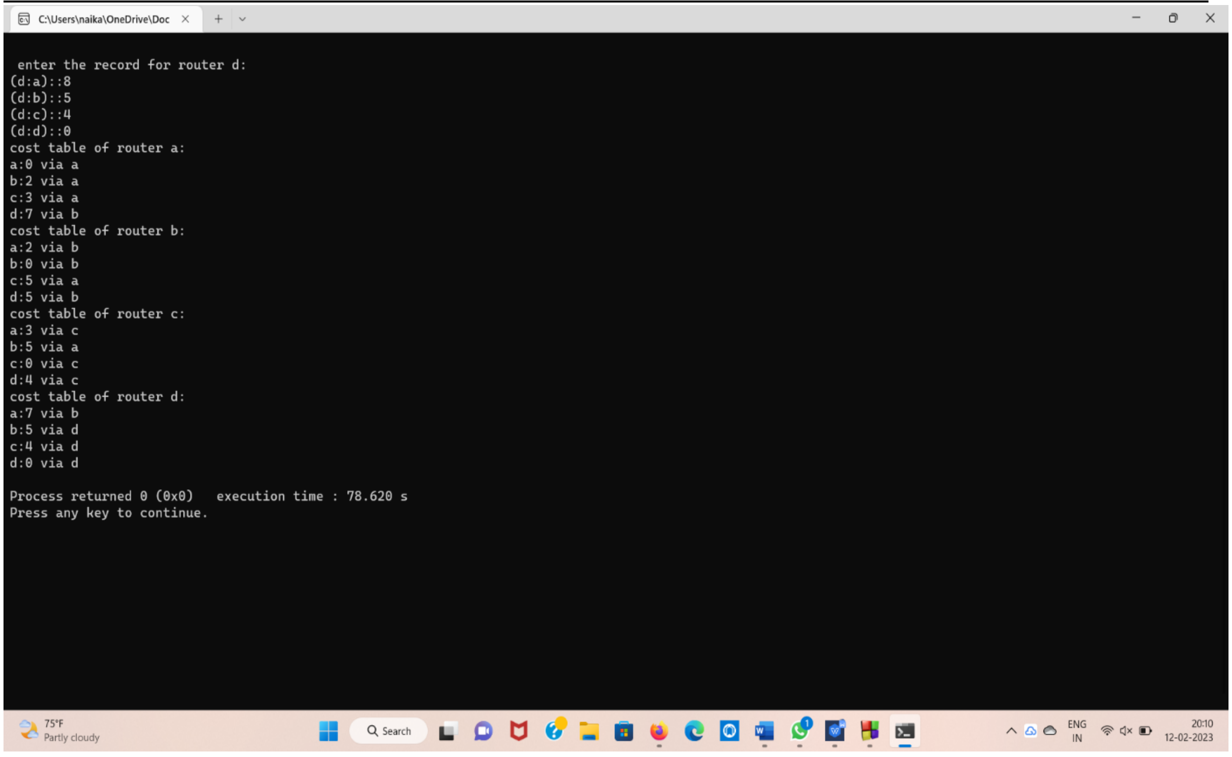
```
        for(j=0;j<n;j++)
                temp[i][j]=g[i][j];
        for(i=0;i<n;i++)
        for(j=0;j<n;j++)
        for(k=0;k<n;k++)
                via[i][k]=i;
        do
{

          count=0;
          for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                                for(k=0;k<n;k++)
                                        if(g[i][j]+g[j][k]<g[i][k])
                                        {
                                                g[i][k]=g[i][j]+g[j][k];
                                                via[i][k]=j;
                                                count++;
                                        }

}while(count!=0);
        for(i=0;i<n;i++)
        {
                printf("cost table of router %c:\n",i+97);
                for(j=0;j<n;j++)
```

```
                        printf("%c:%d via %c \n",j+97,g[i][j],via[i][j]+97);

            }

}
```

OUTPUT :



```
'c' is not recognized as an internal or external command,
operable program or batch file.

 enter the number of nodes:4

 enter the record for router a:
(a:a)::0
(a:b)::2
(a:c)::3
(a:d)::8

 enter the record for router b:
(b:a)::2
(b:b)::0
(b:c)::999
(b:d)::5

 enter the record for router c:
(c:a)::3
(c:b)::999
(c:c)::0
(c:d)::4

 enter the record for router d:
(d:a)::8
(d:b)::5
(d:c)::4
(d:d)::0
cost table of router a:
a:0 via a
b:2 via a
c:3 via a
d:7 via b
cost table of router b:
a:2 via b
b:0 via b
c:5 via a
d:5 via b
cost table of router c:
a:3 via c
```



```
 enter the record for router d:
(d:a)::8
(d:b)::5
(d:c)::4
(d:d)::0
cost table of router a:
a:0 via a
b:2 via a
c:3 via a
d:7 via b
cost table of router b:
a:2 via b
b:0 via b
c:5 via a
d:5 via b
cost table of router c:
a:3 via c
b:5 via a
c:0 via c
d:4 via c
cost table of router d:
a:7 via b
b:5 via d
c:4 via d
d:0 via d

Process returned 0 (0x0)   execution time : 78.620 s
Press any key to continue.
```

Program No: 05:

Simulate a three nodes point-to-point network with duplex links between them. Set the queue size vary the bandwidth and find the number of packets dropped.

Step 1: Select the Experiment number 13 of the simulator to conduct the above experiment.

Step 2: Draw the suitable network topology on the workspace given in the simulator by drag and drop method.

Step 3: Connect the CPE to node and node to node by click on the numbers given on the CPE and node simultaneously.

Step 4: Select the properties of the CPE by right clicking the CPE. Select traffic generator and fill-up the properties like destination, pumping rate, pay load, priority.

Step 5: Select the properties of the link by right clicking on the link and set the properties like Error rate, Physical media. Bandwidth is varies by varying the physical media.

Step 6: Select the properties for the node by right clicking the node. The properties like queue size, protocol, scheduling techniques are to be set.

Step 7: Configure the network by click on the configure button and then click on simulate button.

Step 8: Experiment is saved by user defined file and obtain the performance characteristics like Number of packets drop.

Step 9: Repeat the above steps for another sample.

Step 10:Click on the analysis button and draw the graph for packet generated v/s throughput.

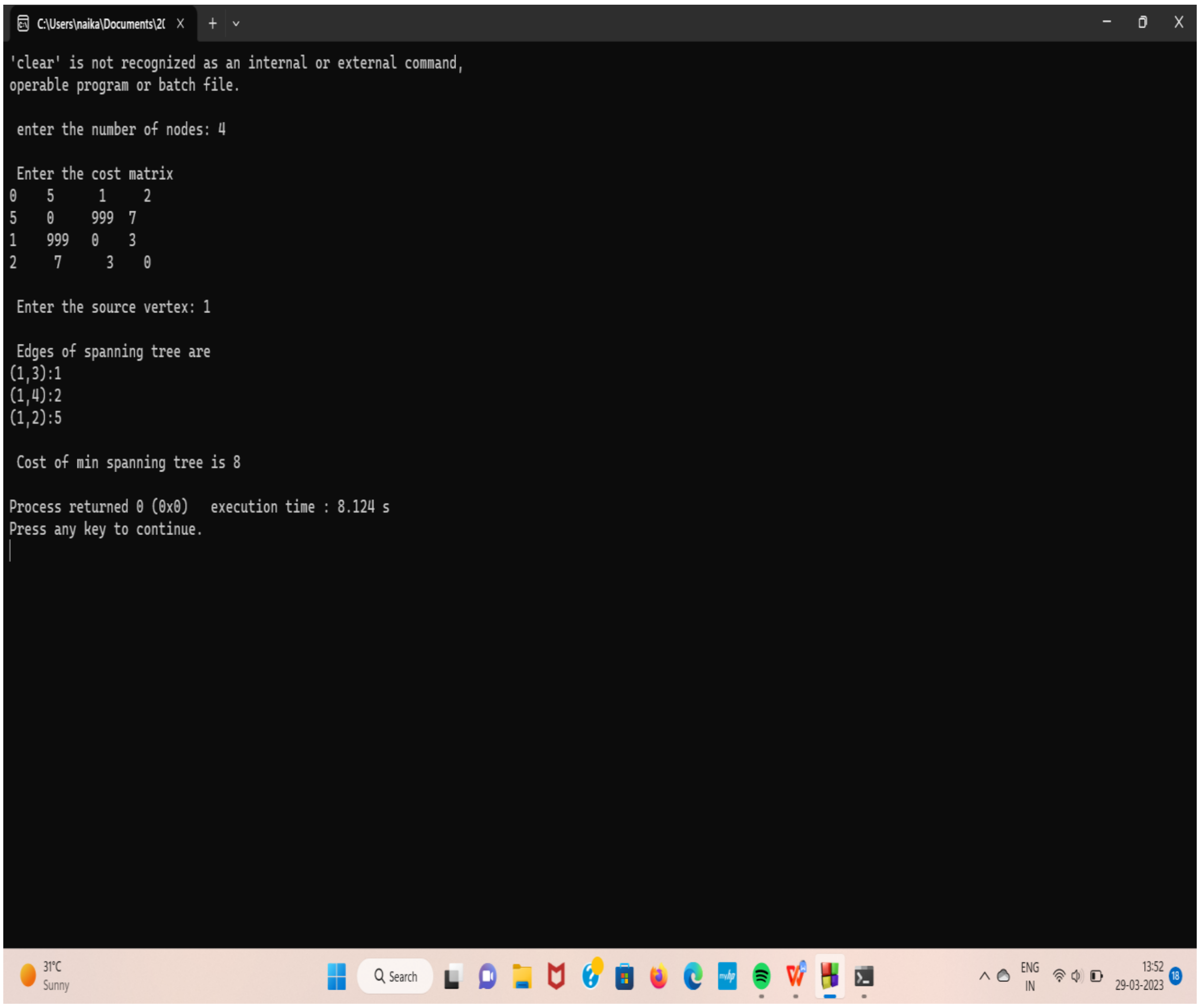
Program No: 06:

Write a program for spanning tree algorithm (Kruskal"s/Prim"s) to find loop less path.

```
#include<stdio.h>
void read_matrix(int cost[10][10],int n)
{
	int i,j;
	for(i=1;i<=n;i++)
	for(j=1;j<=n;j++)
	{
//		printf("(%d,%d):",i,j);
		scanf("%d",&cost[i][j]);
	}
}
void prims(int cost[10][10],int source,int n)
{
	int i,j,sum=0;
	int visited[10],weight[10],vertex[10];
	int min,u,v;
	printf("\n Edges of spanning tree are \n");
	for(i=1;i<=n;i++)
	{
		visited[i]=0;
		vertex[i]=source;
		weight[i]=cost[source][i];
}
visited[source]=1;
for(i=1;i<=n-1;i++)
{
        min=999;
        for(j=1;j<=n;j++)
        {
                if(visited[j]==0 && weight[j]<min)
                {
                        min=weight[j];
                        u=j;
                }
        }
        visited[u]=1;
        sum+=weight[u];
        printf("(%d,%d):%d\n",vertex[u],u,weight[u]);
        for(v=1;v<=n;v++)
        if(visited[v]==0 && cost[u][v]<weight[v])
        {
                weight[v]=cost[u][v];
                vertex[v]=u;
        }
}
printf("\n Cost of min spanning tree is %d\n ",sum);
if(sum>=999)
```

```
	printf("\nSpanning tree does not exist \n");

}

int main()

{

	int n,cost[10][10],source;

	system("clear");

	printf("\n enter the number of nodes: ");

	scanf("%d",&n);

	printf("\n Enter the cost matrix \n");

	read_matrix(cost,n);

	printf("\n Enter the source vertex: ");

	scanf("%d",&source);

	prims(cost,source,n);

}
```

OUTPUT :



```
'clear' is not recognized as an internal or external command,
operable program or batch file.

 enter the number of nodes: 4

 Enter the cost matrix
0    5      1     2
5    0     999  7
1    999   0    3
2     7      3    0

 Enter the source vertex: 1

 Edges of spanning tree are
(1,3):1
(1,4):2
(1,2):5

 Cost of min spanning tree is 8

Process returned 0 (0x0)   execution time : 8.124 s
Press any key to continue.
```

Program No: 07:

Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

SERVER

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<errno.h>

#include<string.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<fcntl.h>

#include<netinet/in.h>

#include<sys/socket.h>

#include<arpa/inet.h>

#include<sys/wait.h>

#include<signal.h>

#define MYPORT 6490

#define BACKLOG 10

int main(void)

{

        int sockfd,fp,new_fd;

        struct sockaddr_in my_addr,their_addr;
```

```
int sin_size,i=0;

int yes=1;

char buf1[20],buf2[20000];

if((sockfd=socket(AF_INET,SOCK_STREAM,0))==-1)

{

	perror("socket");

	exit(1);

}

if(setsockopt(sockfd,SOL_SOCKET,SO_REUSEADDR,&yes,sizeof(int))==-1)

{

	perror("setsockopt");

	exit(1);

}

my_addr.sin_family=AF_INET;

my_addr.sin_port=htons(MYPORT);

my_addr.sin_addr.s_addr=INADDR_ANY;

memset(&(my_addr.sin_zero), '\0', 8);

if(bind(sockfd,(struct sockaddr *)&my_addr, sizeof(struct sockaddr)) ==-1)

{

	perror("Bind");

	exit(1);

}

if(listen(sockfd, BACKLOG) == -1)

{

	perror("listen");
```

```
        exit(1);
}
printf("\n SERVER is online! \n SERVER: Waiting for the client........\n");
sin_size=sizeof(struct sockaddr_in);
if((new_fd=accept(sockfd,(struct sockaddr *)&their_addr, &sin_size))==-1)
{
        perror("Accept");
        exit(0);
}
printf("\n SERVER: Got connection from %s \n", inet_ntoa(their_addr.sin_addr));
recv(new_fd,&buf1,sizeof(buf1),0);
printf("File requested is %s\n", buf1);
if((fp=open(buf1,O_RDONLY))<0)
{
        printf("File not found\n");
        strcpy(buf2,"File not found");
}
else
{
        printf("SERVER: %s found and ready to transfer.\n",buf1);
        read(fp,&buf2,20000);
        close(fp);
}

send(new_fd,&buf2,sizeof(buf2),0);
```

```
        close(new_fd);

        close(sockfd);

        printf("Transfer success \n");

        printf("\n");

        return 0;

}
```

CLIENT

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<errno.h>

#include<string.h>

#include<netdb.h>

#include<sys/types.h>

#include<netinet/in.h>

#include<sys/socket.h>

#include<fcntl.h>

#define PORT 6490

int main()

{

        int i=0,sockfd,len;

        char buf1[40],buf2[20000];

        FILE* fp;
```

```
struct sockaddr_in their_addr;


if((sockfd=socket(AF_INET, SOCK_STREAM,0))==-1)
{
        perror("socket");
        exit(1);
}
their_addr.sin_family=AF_INET;
their_addr.sin_port=htons(PORT);
their_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
memset(&(their_addr.sin_zero), '\0', 8);
if(connect(sockfd, (struct sockaddr *)&their_addr, sizeof(struct sockaddr))==-1)
{
        perror("connect");
        exit(1);
}
printf("CLIENT is online!\n");
printf("CLIENT:Enter the filename to be displayed: ");
scanf("%s",buf1);
send(sockfd,buf1,sizeof(buf1),0);
if(recv(sockfd,buf2,sizeof(buf2),0)==1)
{
        perror("recv");
        exit(1);
}
```

```
	else

	{

		printf("Displyaing the contents of %s",buf1);

		printf("\n%s\n",buf2);

	}


	close(sockfd);

	return 0;

}
```

Program No: 08:

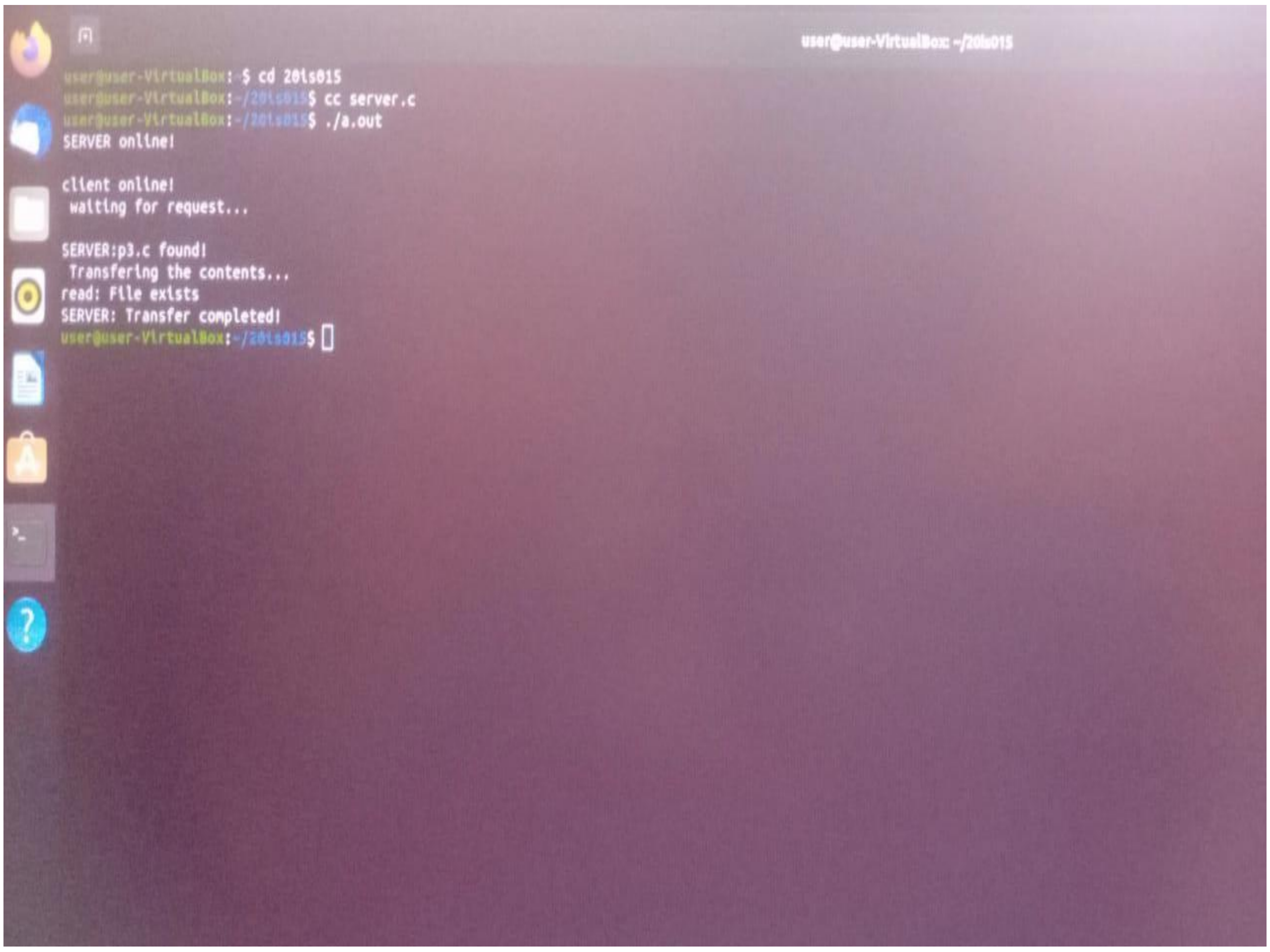
Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.

SERVER

```
#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#define FIFO1_NAME "Server_fifo"

#define FIFO2_NAME "Client_fifo"

int main()

{

     char p[100],c[300];

      int num,num2,fl,fd,fd2;

      mknod(FIFO1_NAME, S_IFIFO | 0666, 0);

      mknod(FIFO2_NAME, S_IFIFO | 0666, 0);

      printf("SERVER Online!\n\n");

      fd = open(FIFO1_NAME, O_RDONLY);

      printf("Client Online!\n Waiting for request...\n\n");

      while(1)

      {
```

```
if ((num = read(fd, p, 100)) == -1)

perror("read error\n");

else

          {

p[num] = '\0';

if((fl=open(p,O_RDONLY))<0)

{

          printf("\nSERVER: %s not found!\n",p);

          exit(1);

}

                    else

{

printf("SERVER: %s found!\nTransfering the contents...\n ",p);

stdin=fdopen(fl,"r");

                    while(!feof(stdin)){

if(fgets(c,300,stdin)!=NULL)

{

          fd2=open(FIFO2_NAME, O_WRONLY);

          if(num2=write(fd2,c,strlen(c))==-1)

          perror("Tranfer error\n");


                     }

else

                              perror("read ");
                    }

                    printf("SERVER: Transfer Completed!\n");

      exit(1);

             }

      }

  }

  return 1;

}
```

OUTPUT :

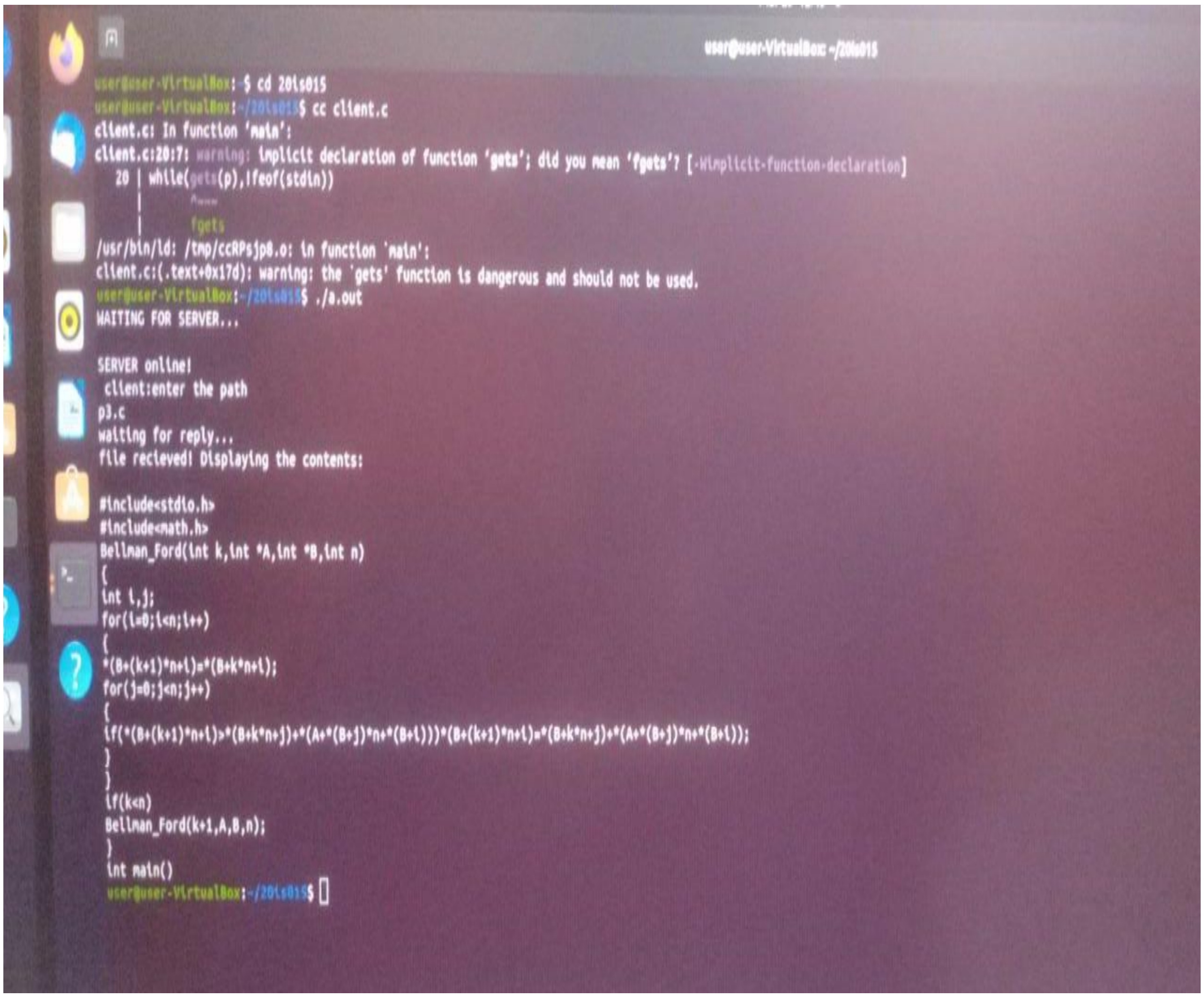


```
user@user-VirtualBox:~$ cd 20is015
user@user-VirtualBox:~/20is015$ cc server.c
user@user-VirtualBox:~/20is015$ ./a.out
SERVER online!

client online!
 waiting for request...

SERVER:p3.c found!
 Transfering the contents...
read: File exists
SERVER: Transfer completed!
user@user-VirtualBox:~/20is015$ 
```

CLIENT

```
#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#define FIFO1_NAME "Server_fifo"

#define FIFO2_NAME "Client_fifo"

int main()

{

            char p[100],f[100],c[300];

        int num,num2,fl,fd,fd2;

        mknod(FIFO1_NAME, S_IFIFO | 0666, 0);

        mknod(FIFO2_NAME, S_IFIFO | 0666, 0);

        printf("\nWaiting for SERVER...\n");

        fd = open(FIFO1_NAME, O_WRONLY);

        printf("\nSERVER Online!\nClient: Enter the path\n");

        while (gets(p) , !feof(stdin))

            {

            if ((num = write(fd, p, strlen(p))) == -1)

            perror("write error\n");

            else
        {
        printf("Waiting for reply...\n");
        fd2 =open(FIFO2_NAME , O_RDONLY);
        if((num2=read(fd2,c,300))==-1)
                perror("Transfer error!\n");
        else
        {
                printf("File recieved! Displaying the contents:\n\n");
                                if(fputs(c,stdout)==EOF)
                perror("print error\n");
                exit(1);
        }
           }
        }
 return 1;
}
```

OUTPUT :



```
user@user-VirtualBox: ~/20is015

user@user-VirtualBox:~$ cd 20is015
user@user-VirtualBox:~/20is015$ cc client.c
client.c: In function 'main':
client.c:20:7: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-function-declaration]
   20 | while(gets(p),!feof(stdin))
      |       ^~~~
      |       fgets
/usr/bin/ld: /tmp/ccRPsjp8.o: in function 'main':
client.c:(.text+0x17d): warning: the `gets' function is dangerous and should not be used.
user@user-VirtualBox:~/20is015$ ./a.out
WAITING FOR SERVER...

SERVER online!
 client:enter the path
p3.c
waiting for reply...
file recieved! Displaying the contents:

#include<stdio.h>
#include<math.h>
Bellman_Ford(int k,int *A,int *B,int n)
{
int i,j;
for(i=0;i<n;i++)
{
*(B+(k+1)*n+i)=*(B+k*n+i);
for(j=0;j<n;j++)
{
if(*(B+(k+1)*n+i)>*(B+k*n+j)+*(A+*(B+j)*n+*(B+i)))*(B+(k+1)*n+i)=*(B+k*n+j)+*(A+*(B+j)*n+*(B+i));
}
}
if(k<n)
Bellman_Ford(k+1,A,B,n);
}
int main()
user@user-VirtualBox:~/20is015$ 
```

Program No: 09:

Write a program to demonstrate HDLC frame to perform bit stuffing.

```
#include<stdio.h>
#include<string.h>
int main()
{
   int a[20],b[30],i,j,k,count,n;
   printf("Enter frame size:");
   scanf("%d",&n);
   printf("Enter the frame in the form of 0 and 1 :");
   for(i=0; i<n; i++)
      scanf("%d",&a[i]);
   i=0;
   count=1;
   j=0;
   while(i<n)
   {
      if(a[i]==1)
      {
         b[j]=a[i];
         for(k=i+1; a[k]==1 && k<n && count<5; k++)
         {
            j++;
            b[j]=a[k];
            count++;
            if(count==5)
            {
               j++;
               b[j]=0;
            }
            i=k;
```

```
        }

         }
         else
         {
             b[j]=a[i];
         }
         i++;
         j++;
     }
     printf("After Bit Stuffing :");
     for(i=0; i<j; i++)
         printf("%d",b[i]);
     return 0;
 }
```

Program No: 10:

10.Simulate a four node point-to-point network, and connect the links as follows: n0-n2, n1-n2 and n2-n3. Apply TCP agent between n0-n3 and UDP n1-n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets by TCP/UDP.

Step 1: Select the Experiment number 14 of the simulator to conduct the above experiment.

Step 2: Draw the suitable network topology on the workspace given in the problem statement (connect CPE1 – n1, n1 – n3, n3 – n4, n4 - receiving point apply TCP for this connection. Connect CPE2 – node2, n2 - n4 apply UDP by drag and drop method.)

Step 3: Connect the CPE to node and node to node by click on the numbers given on the CPE and node simultaneously.

Step 4: Select the properties of the CPE1 by right clicking the CPE1. Select traffic generator and fill-up the properties like duration, window size. Select the properties of the CPE2 by right clicking the CPE2. Apply UDP

Step 5: Select the properties of the link by right clicking on the link and set the properties like Error rate, Physical media. Bandwidth is varies by varying the physical media.

Step 6: Select the properties for the node by right clicking the node. The properties like buffer size is to be set.

Step 7: Configure the network by click on the configure button and then click on simulate button.

Step 8: Experiment is saved by user-defined file and obtain the performance characteristics like Number of packets dropped in TCP protocol and UDP. The packets dropped in UDP is more compare to TCP. So the TCP is more reliable than UDP.

Step 9: Repeat the above steps for another sample.

Step 10:Click on the analysis button and draw the graph for TCP and UDP packet generated v/s reliability.

<u>Program No: 11:</u>

Write a program to demonstrate hamming code for error correction.

```
#include <math.h>

#include <stdio.h>

// Store input bits

int input[32];

// Store hamming code

int code[32];

int ham_calc(int, int);

void solve(int input[], int);

// Function to calculate bit for

// ith position

int ham_calc(int position, int c_l)

{

    int count = 0, i, j;

    i = position - 1;

    // Traverse to store Hamming Code

    while (i < c_l) {

        for (j = i; j < i + position; j++) {

            // If current boit is 1

            if (code[j] == 1)

                count++;

        }
        // Update i
        i = i + 2 * position;
    }
    if (count % 2 == 0)
        return 0;
    else
        return 1;
}
// Function to calculate hamming code
void solve(int input[], int n)
{
    int i, p_n = 0, c_l, j, k;
    i = 0;
    // Find msg bits having set bit
    // at x'th position of number
    while (n > (int)pow(2, i) - (i + 1)) {
        p_n++;
        i++;
    }
    c_l = p_n + n;
    j = k = 0;
    // Traverse the msgBits
    for (i = 0; i < c_l; i++) {
        // Update the code
        if (i == ((int)pow(2, k) - 1)) {
```

```
        code[i] = 0;
        k++;
    }
    // Update the code[i] to the
    // input character at index j
    else {
        code[i] = input[j];
        j++;
    }
}
// Traverse and update the
// hamming code
for (i = 0; i < p_n; i++) {
    // Find current position
    int position = (int)pow(2, i);
    // Find value at current position
    int value = ham_calc(position, c_l);
    // Update the code
    code[position - 1] = value;
}
// Print the Hamming Code
printf("\nThe generated Code Word is: ");
for (i = 0; i < c_l; i++) {
    printf("%d", code[i]);
}
```

```
}
// Driver Code
void main()
{
    // Given input message Bit
    input[0] = 0;
    input[1] = 1;
    input[2] = 1;
    input[3] = 1;
    int N = 4;
    // Function Call
    solve(input, N);
}
```

Program No: 12:

Write a program for congestion control using Leaky bucket algorithm.

```
#include<stdio.h>

struct frame

{          char msg[20];

           int seq;

}fr[40];

int f_no,front=0,rear=-1,count=0,q_size;

char q[10][20];

void insert(int i)

{

          if(count==q_size)

          {

                    printf("\n bucket is full\n");

                    ("\n packet lost is %s\n",fr[i].msg);

                    sleep(4);

                    return;

          }

          rear=(rear+1)%q_size;

          strcpy(q[rear],fr[i].msg);

          count++;

          printf("\n inserted message into the bucket is %s\n",fr[i].msg);

          sleep(5);

}
```

```
void del()
{
	if(count==0)
	{
		printf("\n bucket is empty\n");
		return;
	}
	printf("\n deleted message is %s\n",q[front]);
	sleep(5);
	front=(front+1)%q_size;
	count--;
}
int main()
{
	int i,j,k,arrival[40],clk=0,n;
	char str[100];
	printf("\n enter the message: ");
	scanf("%s",str);
	printf("\n enter the queue size: ");
	scanf("%d",&q_size);
	for(i=0;str[i]!='\0';)
	{
		fr[f_no].seq=f_no;
		for(j=i,k=0;str[j]!='\0' && k<5;j++,k++)
			fr[f_no].msg[k]=str[j];
```

```
        fr[f_no].msg[k]='\0';
        f_no++;
        i=j;
}
printf("\n show the packets: \n");
for(i=0;i<f_no;i++)
        printf("frame %d is : %s\n",i,fr[i].msg);
sleep(5);
arrival[0]=1;
for(i=1;i<f_no;i++)
{
        n=rand()%6;
        arrival[i]=arrival[i-1]+n;
}
printf("\n arrival time of the packets\n");
for(i=0;i<f_no;i++)
        printf("frame %d : %d \n",i,arrival[i]);
printf("size of the bucket is %d\n",q_size);
sleep(5);
i=0;
while(i<f_no||clk<=5*f_no)
{
        printf("\nclk:: %d",clk);
        while(clk==arrival[i])
        {
```

```
                        insert(i);

                        i++;

                }

                if((clk%5)==0)

                        del();

                clk++;

        }

}
```

Program No: 13:

13. Simulate the different types of Internet traffic such as FTP a TELNET over a network and analyze the throughput.

Step 1: Select the Experiment number 16 of the simulator to conduct the above experiment.

Step 2: Draw the suitable network topology on the workspace given in the simulator.

Step 3: Connect the CPE to node and node to node by click on the numbers given on the CPE and node simultaneously.

Step 4: Select the properties of the CPE1 by right clicking the CPE1. Select traffic generator and fill-up the properties like destination point, application type (FTP/TELNET), Duration in seconds, window size.

Step 5: Select the properties of the link by right clicking on the link and set the properties like Error rate, Physical media. Bandwidth is varies by varying the physical media.

Step 6: Select the properties for the node by right clicking the node. The properties like buffer size is to be set.

Step 7: Configure the network by click on the configure button and then click on simulate button.

Step 8: Experiment is saved by user-defined file and obtain the performance characteristics.

Step 9: Repeat the experiment by selecting suitable application type FTP and TELNET.

Step 10: Check the performance characteristics by applying TELNET for all the CPE's then check the performance characteristics by applying FTP.

Step 11: Repeat the above steps for another sample.

Step 12:Click on the analysis button and draw the graph for protocol v/s reliability.

Program No:14:

14. Simulate an Ethernet LAN using N-nodes (6-10), change error rate and data rate and compare the throughput.

Step 1: Select the Experiment number 3 of the simulator to conduct the above experiment.

Step 2: Draw the suitable network topology on the workspace given in the simulator by drag and drop method.

Step 3: Select the properties of the node by right clicking the node. Select the properties like transmission type, Payload.

Step 4: Connect the Hub to Hub by right clicking the 'U' button, select add connection, choose the port number and then click connect button. Select the properties of the Hub by right clicking the 'U' button and set the properties on data rate, error rate.

Step 5: Click on the configure button and then click on simulate button.

Step 6: Experiment is saved by user-defined file and obtain the performance characteristics.

Step 7: Repeat the experiment by conducting the above steps.

Step 8: Check the performance characteristics.

Step 9: Repeat the above steps for another sample

Step 10: Click on the analysis button and draw the graph for Error rate v/s throughput, data rate v/s throughput.